Python和人工智能基础课程（第四课）

雷萧萧，张威

上期复习

- 逻辑运算
- 成员运算
- 身份运算
- 运算优先级
- 流程控制
  - if; if else; if elif else
  - loop: while; for
  - break; continue; pass

练习讲解

1. 创建一个list，每回在升序排列中insert一个数字，最多10个。不可以用list.sort()

2 找到100以内最大的质数

3 找到最长的substring

今日摘要

- 函数function（方法）
- 函数的定义
- 函数的参数
  - 必备参数
  - 关键字参数
  - 缺省参数
  - 不定长参数
- 函数的返回值
- 匿名函数
- 变量作用域
- 模块

定义函数

- 函数的定义通过以下代码块完成

```
def function_name (parameters ):
        function_suite
        return [expression]
```

函数名称

- def function_name (parameter1, parameter2, ...) :
- def是python关键字，表示定义一个函数
- function_name是用户自定义的函数名称，要和变量起名一样要简洁，有描述性
- 括号中为用户自定义参数，参数作为函数的输入(input)为函数提供数据进行操作
- 函数可以没有参数，可以有1至n个明确数量的参数，也可以有不定数量的参数
- 记住最后要有冒号“: ”

函数主体

- 函数主体需要缩进
- 函数主体由n行代码组成
- N行代码执行某个任务
- 任务可以有参数，可以没有参数
- return语句标志着函数的结束

```
return some_variable
```

- 函数主体最后会有return语句，表示将某个结果（output）返回给使用者
- 没有return语句的函数，相当于返回None
- 函数主题是函数子内容，需要缩进；后面的代码和def是平级，保持平行，不需要缩进

```
def my_function ( ) :
        statements
        return variable
next_code...
```

- 黑箱概念

函数主体

- 可以在函数主体中定义局部变量

```
def int_multiplication (num1, num2) :
        result = num1 * num2
        return result
print int_multiplication (10, 20)
```

- 这个例子中，function有两行代码
- 计算了两个数字相乘的结果
- 返回计算值

调用函数

- 用户通过输入函数名称，加上括号（ ），并给括号中一定参数（输入）来调用函数
- 回想：numpy.array(), list.sort()
- 函数要先定义，再使用

```
def int_multiplication (num1, num2) :
        result = num1 * num2
        return result
print int_multiplication (10, 20)
```

- 计算机按一行一行的顺序执行代码
- 那到函数定义的地方执行什么了呢?
- 函数定义时，计算机会在内存中保存一个模板“template”，而并没有任何实际的计算操作
  只有到真正调用函数时，才会进行函数主体内部的操作

调用函数

- 现在我们定义以下函数

```
def money_for_girlfriend (my_income ) :
        money4gf = my_income*0.8
return money4gf
```

- 以下3中调用方式有什么不同?

```
my_salary = 8000
print money_for_girlfriend (my_salary)

my_salary = 8000
to_girlfriend = money_for_girlfriend (my_salary)
print to_girlfriend

print money_for_girlfriend (8000)
```

参数

- 函数在定义和调用时，函数名括号（）中的变量叫参数
- 参数可以是0个，sort()
- 参数也可以是1至多个
- 0个参数的函数

```
def print_hello ( ) :
        print “Hello World!”
        return
```

  - 0个参数的函数的输入来源于函数内部
- 一个参数的函数

```
def add_welcome (guest_name) :
        str = "Respectful Mr." + guest_name + ", Welcome to Python"
return str
user_name = input ( )
print add_welcome (user_name)
```

参数

- 两个参数的函数

```
def two_int_addition (int1, int2) :
        result = int1 + int2
        return result
num1 = 10
num2 = 20
print two_int_addition (num1, num2)
```

- 函数的参数数量也可以是N个

形式参数（形参）

- 函数定义时括号中的参数叫形式参数（形参）
- 我们来看一下以前几个例子的函数定义

```
def add_welcome (guest_name) :
def two_int_addition (int1, int2) :
def two_int_addition (int1, int2) :
```

- 其中的guest_name, user_name, int1,int2叫做形参
- 这些参数作为输入，由函数主体对它们进行一些操作
- 函数主体中对这些参数使用时，要用一样的名称guest_name -> guest_name

```
def add_welcome (guest_name) :
        str = "Respectful Mr." + guest_name + ", Welcome to Python”
        return str
user_name = input ( )
 print add_welcome (user_name)
```

实际参数（实参）

- 函数调用时在括号里填入的参数，叫实参（实际参数）。
- 可以理解为在函数实际使用时的真正参数
- 实参可以是一个具体的数据，比如

  print add_welcome ("Kobe”)

- 实参也可以是一个对变量的引用

  my_name = "Kobe”

  print add_welcome (my_name)

- 在实际工作中，实参基本都是对变量的引用
- 调用函数时，括号里实参变量会自动根据顺序对应到函数定义时的形参上

  print two_int_addition (10, 20)

  10会对应到num1

  20会对应到num2

必备参数

- 如果函数定义时没有参数，使用时括号中也不必加任何内容，如int( )
- 如果函数定义时有形参，那在调用函数时必须在括号中输入匹配数量的参数
- 没有参数或数量不对，都会报错

```
print add_welcome ( )
print two_int_addition(20)
print two_int_addition(10, 20, 30)
```

- 思考

```
print two_int_addition(“Kobe”, “Smith”)会出现什么结果?
print two_int_addition("Kobe", "24")
```

关键字参数

- python使用关键字参数，使得用户调用函数时不必按函数定义中形参的顺序给函数输入值，而是对形参指定输入值
- 普通情况

```
def name_age (name, age)
        my_name = “My name is ” + name + “.”
        my_age = "My age is " + age + ".”
        return my_name + " " + my_age
print name_age ("Kobe", "24)
```

- 关键字参数
- 函数的定义没有任何不同
- 使用时在括号中明确指定每个参数的值是多少，而不必遵循定义函数时的顺序

```
print name_age (age = 24, name ="Kobe”)
```

- 当然用户可以按原来的顺序，不过没什么必要了

```
print name_age (name = "Kobe", age = 24)
```

缺省参数

- 当函数没有某个参数的输入时，缺省参数会作为默认值做为输入

```
def printinfo(name, age = 35 ):
        print "Name: ", name
        print "Age ", age
         return

printinfo(age=50, name="miki" )
printinfo(name="miki" )
```

不定长参数

- 有时在定义函数时没想好需要用几个参数，可以使用不定长参数

  def functionname([parameters,] *var_args_tuple):

- 例

```
def printinfo( arg1, *vartuple ):
        print "输出: "
        print arg1
        for var in vartuple:
                print var
        return
printinfo( 10 )
printinfo( 70, 60, 50 );
```

函数的返回值

- return语句将某个变量作为返回值/输出值返回给函数的调用者

  return some_variable

- 返回值可以是一个任意类型的变量，也可以是一个数值，也可以返回空值

  return my_money

  return “A”

  return

返回的不同情况

- return可以在函数的结尾

```
def some_function (parameter1, parater2, ...) :
    some_statement
    return some_variable
```

- return标志着函数的结束，但不一定在函数的结尾
- 一个函数也可以有多个return语句

```
def return_grade (score) :
    if score > 80 :
        grade = “A”
        return grade
    else :
        grade = “B”
        return grade
```

匿名函数/lamda函数

- Python使用lamda关键字创建匿名函数
- lamda函数主体是一个表达式（一行），而不是一个代码块（n行）
- 优点，简洁高效； 缺点，不能表达复杂结构
- lambda函数拥有自己的命名空间，且不能访问自有参数列表之外或全局命名空间里的参数
- 使用方法

  lambda [arg1 [,arg2,.....argn]]:expression
- 例

```
add = lambda arg1, arg2: arg1 + arg2

print "相加后的值为 : ", add( 10, 20 )
print "相加后的值为 : ", add( 20, 20 )
```

变量作用域

- 变量在它的作用域内可以被访问/赋值，在作用域外则不可
- 以作用域划分，变量可分为全局变量和局部变量
- 函数内部的变量叫做局部变量，而函数以外的变量叫做全局变量
- 局部变量只能在它被声明的函数内部被访问，A函数中的x变量只能在A函数中范文
- 在函数内部无法改变全局变量的值

  如果局部变量与全局变量重名，在函数内部给局部变量复制仍然无法改变全局变量
- 例

```
x = 1
def change():
        x = 5
        print "x: ", x
        return change ()
print "x: ", x
```

模块（Module）的定义

- Python模块（module）是一个python文件，以.py结尾
- 模块中包含了一个或多个函数（function)
- 调用模块中的函数和直接在代码中写函数有什么区别？函数有什么用?
- 模块让完成特定任务的代码具有更好的封装性，易于复用
- 可以自己复用，也可以开源贡献给其他用户，整个网络社区

定义模块

- 以下为welcome_user.py文件内容，也是一个模块

```
def wel_py (user_name) :
        print "Hello, " + user_name + "!”
        return
```

- 既然模块中包含的是一个函数，那它的内容就不再赘述

调用模块

- 在一个脚本文件中，输入import module_name对模块进行调用

```
import numpy
import scipy
```

- 注意import后面有空格
- 一个模块只会被导入一次
- 当代码执行时，计算机会根据当前的路径搜索该模块

from module import function

- 如果只想从某模块文件中导入某一个函数，而不是整个模块文件，可以用到
  from numpy import sklearn

- 当然，可以从一个模块中导入N个函数，中间以逗号“，”隔开
  from numpy import a, b, c

- 将包括多函数的模块所有内容导入，使用from modname ipmort *
- 注意有空格
  from numpy import *

- 计算机会如何寻找模块呢?
  1. 搜索当前目录
  2. 如果不在当前目录，Python 则搜索在 shell 变量 PYTHONPATH 下的每个目录
  3. 如果都找不到，Python会察看默认路径

练习

1 创建一个函数，接受一个两个数字作为输入，分别打印两个数字，相加，相乘，AND，OR，并打印

2 在一个文件里面创建四个函数，加减乘除，命名为一个新的模块，在另外一个文件中，分别测试调用这四个函数。

3 在一个文件里面创建四个函数，加减乘除，命名为一个新的模块，在另外一个文件中，分别测试调用这四个函数。如果输入不是数字，返回错误提示

4 设计一个函数，给定一个数字，找到比这个数字小的最大的质数。